RESEARCH PLAN PROPOSAL

A New Set of Object Oriented Metrics to Measure Structural Complexity of Software Product Line Feature Model

For registration to the degree of
Doctor of Philosophy

IN THE FACULTY OF SCIENCE

THE IIS UNIVERSITY, JAIPUR

Submitted by

GeetikaVyas
Enroll. No.: ICG/2012/14598

Under the Supervision of

Dr. Amita Sharma
Asst. Professor

Department of Computer Science

March 2013

1. Topic

A new set of Object Oriented Metrics to Measure Structural Complexity of Software Product Line Feature Model

2. Introduction

Designing software for addressing rapidly changing requirements is a challenging task. Nowadays software evolution requires continuous adaption, extension and customization [1]. As a result new software development methodologies and tools continue to evolve to address emerging issues. These approaches share some common behavior and goals: increasing the productivity of the development reams, reducing products time to market, considering future aspects, reducing development costs and improving customer satisfaction [2].

Software product line paradigm is one such software development approach which focuses on reuse which is planned, enabled and obligatory by nature. It aims on strategic reuse leading to business benefits like improved time to market, enhanced performance, reduced development and maintenance costs, mass customization and improved software quality [3].

2.1. Software Product Line:

As defined by Clements and Northrop (2002) Software Product Line is:

"A set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment".

A software product line, or software product line development, includes software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production. It focuses on systematic reuse instead of ad hoc reuse. This reuse is achieved in the case when applications share features or characteristics.

In context of software product lines, a line means a set of software products that are related and share commonalities like data structures, software components and architecture etc [4]. The main concept of software product lines is to prepare and support reuse of core assets and the variability of products in a line, so as to improve time to market, performance, and quality to reduce development cost and effort.

Software product line approach is structured into two main processes [5]: Domain Engineering and Application Engineering. Domain engineering finds, records, constructs and presents set of software artifacts that can be used in future software's which specialize in that particular application domain. As a result, domain knowledge is achieved in the form of reusable software assets, which leads to faster software production and reduced time-to-market. Domain engineering is the basis of software product line practices and is also called engineering-for-reuse whereas application engineering which produces the individual applications is often called engineering-with-reuse. Application engineering centers on development with reuse, and it develops the final products, using the core assets and the specific requirements expressed by the customers.

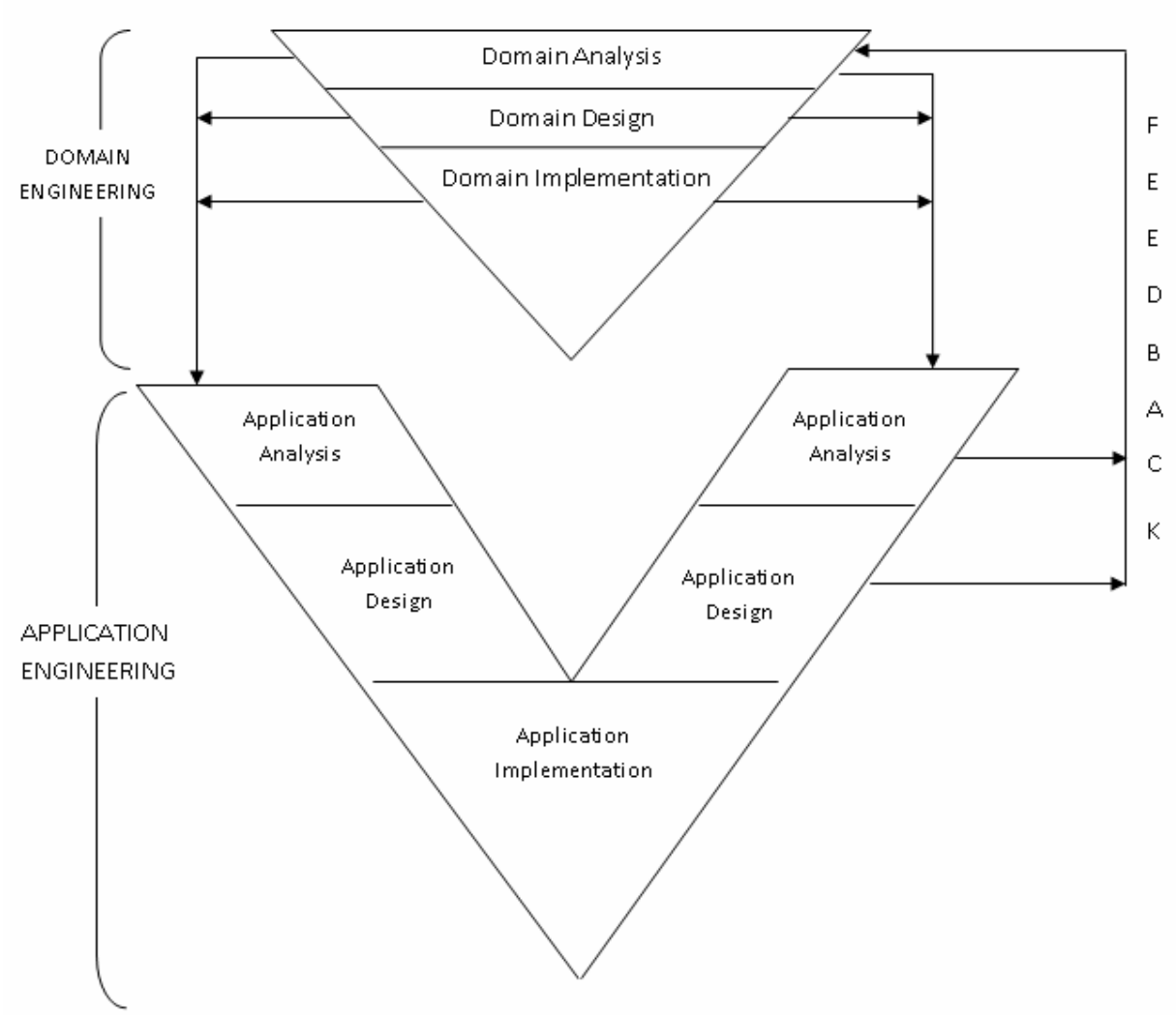

Figure 1 The Lifecycle of Software Product Line.

Software product line is gaining popularity because it aims at the specification, construction, management and maintenance of complex software systems with the goal of reuse, variability, and efficiency. The following benefits can be gained by software product line:

- reduced time-to-market and time-to-revenue for new products
- improved competitive product value
- higher profit margins
- improved ability to hit market windows
- better product quality and improved company reputation for quality
- improved scalability of business model in terms of products and markets
- increased agility to expand into new markets
- reduced risk in product deployments

2.2. Feature Oriented Programming:

Feature Oriented Programming was designed with an eye towards software product line approach [6]. The idea of feature oriented programming is to decompose a software system in terms of the features it provides. It modularizes the software system into feature model which is a representation of a product's features and the dependencies of these features. It is used explicitly for systematically defining the commonality and variability. Sharma et. al [7] recommend this methodology because of its ability to produce numerous similar but functionally different programs from the same set of features. This is achieved by simply selecting the desired features. It helps in constructing well-structured software which can be tailored to the specific user needs and the application scenario. This methodology increases several software quality attributes like adaptability, modularity, traceability, readability, , maintainability, extendibility, understandability reusability, flexibility, and ease of evolution. Thus it is preferred for software product line development.

2.3. Features & Feature Model:

"A feature is a structure that extends and modifies the structure of a given program in order to satisfy a stake holder's requirement, to implement and encapsulate a design decision, and to offer a configuration option".

A feature is a unit of functionality that satisfies a requirement, represents a design decision, and provides a potential configuration option. They are modular entities encapsulating a particular functionality of the system.

They are helpful in the explanation of commonality and variability in the analysis, design, and implementation of software product lines. Classification of features increases the comprehensibility of a software product line. With the selection and removal of features, software provides different facilities and different configurations. Typically, from a set of features, many different software systems can be generated that share some common features and at the same time differ in others.

A feature model represents all the products of the software product line. These feature models are visually represented by feature diagrams. Set of features can be selected or deselected and varied versions of the software can be generated from a single feature model [8].

Hierarchically arranged features of a feature model can be classified as [9]:

- Mandatory- features must be included in the description of the parent feature.
- Alternative feature group- one and only one feature from the group can be included in the description of the parent.
- Optional- the feature may or may not be included in the description of the parent.
- Or feature group- one or more features from a feature group can be included in the description of the parent.
- Excluded- the presence of the feature requires elimination of another feature or set of features.
- Includes- the presence of given feature requires the existence of another feature or set of features.

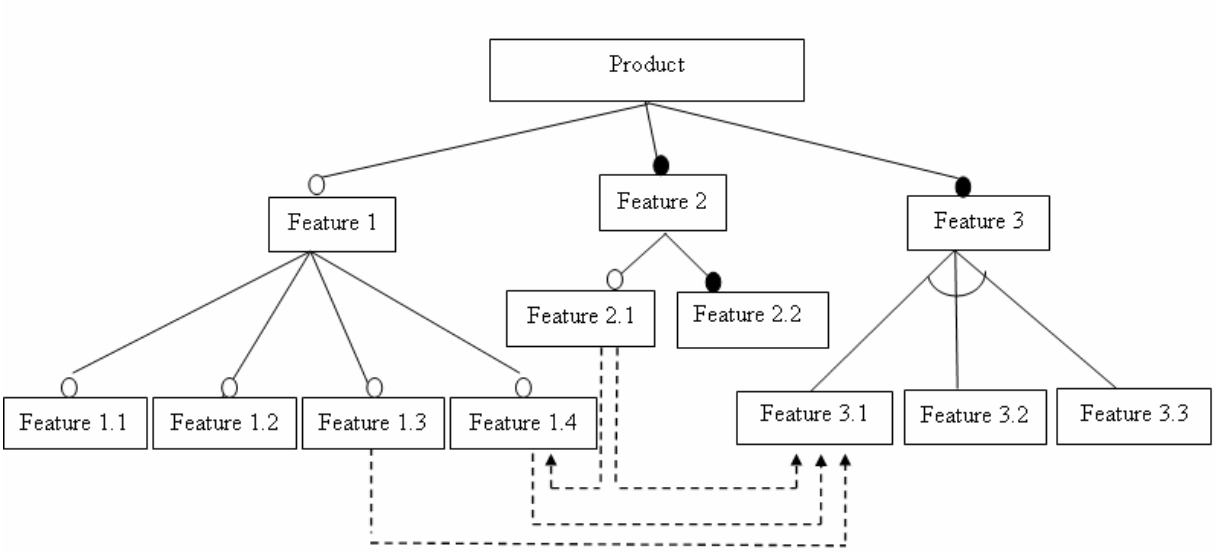

Figure 2 Generic Feature Model.

Figure 2 depicts a generic feature model. It has 3 features numbered feature 1, feature 2 and feature 3 respectively. Out of which feature 2 and feature 3 are mandatory and feature 1is optional. Feature 3 has three alternative features numbered feature 3.1, feature 3.2 and feature 3.3. Feature 1 has four alternative features numbered feature 1.1, feature 1.2, feature 1.3 and feature 1.4. Feature 1.3 and 1.4 features require feature 3.1 to be present mandatory. Feature 2 has one optional feature numbered2.1 and one mandatory feature 2.2. Feature 2.1 requires feature 1.4 and 3.1 to be present mandatory.

2.4. Complexity in Software Product Line Feature Model:

Complexity is defined as the interaction between entities of a system. As the number of entities increases, the number of interactions between them also increases exponentially, leading to increased complexity. In software product line feature models, when features are added, the variability is increased. This in turn increases the complexity of the feature model.

Complexity can be understood in the following ways [10]:

- Problem complexity measures the complexity of the underlying problem.
- Algorithmic complexity reflects the complexity/efficiency of the algorithm implemented to solve the problem.
- Cognitive complexity measures the effort required to understand the software.
- Structural complexity measures the structure of the software used to implement the algorithm (it includes control flow structure, hierarchical structure and modular structure).

Keeping in pace with quickly changing market demands products needs to be more flexible and adaptable. As a result more features are added to the software product line feature model. It leads to increase in size and structural complexity.

Structural complexity is also affected by coupling and cohesion. Therefore high cohesion and low coupling is suggested for a balance in the structural complexity making it easy to operate and manage the software product line.

Measurement of structural complexity helps in outlining those constituents of the structure which make the structure better or worse. Complexity along with variability plays an important role in software product line therefore its assessment is very essential. The assessment of complexity can be done with the help of metrics.

Available measures for software product line feature models:

- Measures of size: Number of Features (NF), Number of Top Features (NTop), Number of Leaf Features (NLeaf).
- Measures of Structural Complexity: Cyclomatic Complexity (CC), Cross-Tree Constraints (CTC), Ratio of Variability (RoV), Coefficient of Connectivity -Density (CoC), Flexibility of Configuration (FoC), and Number of Valid Configurations (NVC).
- Measures of length: depth of tree (DT).

2.5. Metrics:

Metrics are software measurement units, which measure the degree to which a given system, component or process possesses a given attribute. Metrics use numerical ratings to measure various domains like the complexity and reliability of source code, the length and quality of the development process and the performance of the application when completed [10]. Variety of traditional metrics are available for assessing software quality and complexity, but as software product lines have specific characteristics (e.g., variability, commonality, compositionality), these measures to evaluate the quality of individual products cannot be applied easily to evaluate software product lines.

Object-oriented metrics are a set of standards by which one can measure the effectiveness of object-oriented techniques in respect to the design of a system as in the case of software product line paradigm. These metrics support features like localization, encapsulation, information hiding, abstraction and inheritance etc. Software product line approach also possesses these characteristics and therefore the object oriented metrics find an application here.

Despite the emergence of methods and techniques, the need of measures for assessing quality attributes in software product line feature model still needs to be fulfilled [11].This need of metrics is heightened all the more when an organization is adopting an upcoming technology like software product lines for which established practices have yet to be developed and metrics, processes and standards existing in traditional software development cannot be applied straightforward. [12]

3. Review of Literature

J. Bosh et. al [13] state that software product line is an approach which favors increased software reuse and reduced development cost .In this paradigm the architecture and a set of reusable components are shared among a family of products. The author concluded that evolution in software product lines is more intricate than in traditional software development. The reason for this is that new and (at times) contrasting requirements keep originating from the existing products in the product line and also from new products that are to be integrated. The increase in requirements leads to addition of features and this increases the complexity of the product line.

L. Chen et. al [14] have identified and described many issues related to variability and variability modeling in reference to software product line. The authors identified many factors which lead to increase in variability viz. mechanism selection, tool support, implicit dependencies, phase selection, deficiency of feature modeling, unavailability of uniform representation of variability throughout the entire lifecycle, inability to model the details about variation point for re-users to build variants, inability in describing complex dependencies and dependency interactions etc. He also reflected that because of these issues variability increases and management of variability becomes difficult. This increase in variability results in rise in complexity. He concluded that variability management needs attention and measures should be developed for the efficient management of the complexity.

Maben et. al [15] term modeling of variability as a significant task. The author stated that feature modeling is one approach to deal with complexity. But expressing several requirements in features makes feature models very complex. He emphasized over careful development and maintenance of feature models keeping in mind redundancy and consistency. Stating the importance of Feature models he clarified that these models are not only used for domain modeling, but also for product derivation in product line development. And also this that inconsistent feature models limit the chance to build consistent product configurations. Thus complexity should be first measured in feature model and then in the architecture of the product line.

But the traditional tools and measures cannot be adopted, according to Jamali et. al [16] it is premature to apply traditional metrics when there is uncertainty about the precise definitions of many of the quantities to be observed and their impact upon subsequent metrics. The usefulness of such metrics can be enhanced only if clearer guidance concerning their application is provided. Thus research spotlighting on development and implementation of a suite of metrics for OO design is considered highly necessary.

Fenton et. al reflected the need of complexity metrics in their study and stated that complexity metrics are comparatively good indicators of maintainability. Complexity metrics also help to identify modules which are likely to be fault-prone pre-release at a very early stage in the development phase of software product lines. According to the author the drawbacks of LOC (line of code) has led to the derivation of complexity metrics and function points.

According to Kearney et. al [17] if software complexity measures are used inappropriately they will have large, damaging effects leading to poor programming practices and will thus demoralize good programmers. Complexity measures typically depend on factors like control structure, program size, or the nature of module interfaces. These measures have great potential for reducing the cost of software development and maintenance. But the authors felt that development of these measures is difficult because programming behaviors are not understood clearly. The available software complexity measures have not realized their potential for the management and reduction of software cost. This failure leads to the motivation that appropriate complexity measures need to be developed which eventually leads to the measure of structural complexity.

Study shows that very limited work has been done in the field of defining and validating metrics and assessment of the internal and external quality attributes. Only few researchers have contributed in the field of developing quality metrics for software product lines feature models.

Oliveira et. al [18] have proposed a metric suite to support evaluation of software product line architecture (PLA). The focus was to provide an alternate metric suite to perform software product line architecture evaluation based on quality attributes. They also presented an evaluation focusing the product line complexity, to show how the metric suite could support the development of product line architecture. The metrics proposed were found to be architecture centric. Feature models and their complexity holds immense importance towards increasing the life span of the product line and the paper failed at answering issues such as the impact of inclusion/update/removal of a specific feature of a product line, and alternatives for a less complex configuration of a product line.

Asim Rahman [19] proposed metrics to quantify the structural quality of product line architecture. The author concluded that structural soundness of the product line architecture could be represented as collection of various quality attributes of the architecture. These quality attributes in turn depend upon the quality and structural attributes of the components that shape the architecture. Hence, to assess the quality attributes of the components as well as the architecture, a model comprising of a set of metrics is necessary and a single metric is inadequate for the job. Here also the focal point of the author was quality attributes of the product line architecture, whereas feature models were totally sidelined.

Zhang et. al [20] have also proposed metrics for accessing quality of product line architecture. But they are also not defined and considered within the premise of measurement theory. In other words these proposed metrics seem to be rather ad-hoc.

A set of structural metrics have been proposed by Bagheri et.al for assessing the maintainability of software product lines feature model. The authors recognize the need of early indicators of external quality attributes, in order to avoid the implications of defective and low quality design during the late stages of production. Thus a set of structural metrics for software product line feature models are proposed and theoretically validated. Further, they investigated whether the proposed structural metrics were effective early indicators of the three main sub characteristics of maintainability viz. analyzability, changeability, and understandability. But the core focus of software product lines is on reusability and the metrics proposed in the paper were found to be inefficient to assess the same.

Only handful researchers have addressed the issue of proposing appropriate structural quality metrics for software product line feature models. Metrics are for assessing the structural complexity of the software product lines feature model. The requirement of valid metrics and the limitations of the currently available metrics, gives motivation towards research in this direction.

4. Motivation

Structural complexity is a widely known internal factor to measure quality attributes. The structural complexity of feature model affects the overall quality of the software product line. In case of software product lines measuring this complexity by means of feature models seems to be a good approach for predicting, controlling and evaluating the overall complexity in the structure of the software product line.

Literature review shows that there is lack of generalized metrics to measure the structural complexity of software product line feature model. Authors proposing the measures claim that their measures are validated empirically through case study, but the case study is just a proof to show how the measures work. Also majority of these measures are proposed for evaluating the architecture or maintainability of software product line feature model, the remaining quality characteristics being unobserved.

Reusability and reliability are strong pillars of software product line feature model and structural complexity had direct effect on them. But unfortunately these important factors are left untouched. The aim of this research is to study reusability and reliability of software product line feature model and propose generalized metrics for measuring the structural complexity of the same. These measures will help us to analyze the effect of reusability applied to a product line and also this that the reusability is how far reliable. The available measures are domain specific and also fall short in scrutinizing the structural complexity of feature models with respect to reusability and reliability, gives motivation towards research in this direction.

5. Objectives

The major objectives set for the research work are:

1. To study UML model, domain engineering, and software product line feature model.
2. To explore structural metrics (simple and feature oriented)
3. To investigate and analyze existing metrics for measuring structural complexity of feature models.
4. To propose improvement in existing/ new metrics for measuring structural complexity keeping in mind usability (and its two sub characteristics: communicativeness and understandability) of feature models in software product line.
5. To validate the improvement/ new metrics theoretically and through controlled experimentation.
6. To analyze the result and draw conclusions.

6. Plan of work and methodology

6.1. Work Plan:

Phase 1(4-5 months):

Literature survey of UML Model, domain engineering, and software product line feature model.

Phase 2 (2-3 months):

Detail study of structural metrics (including simple and feature oriented metrics both).

Phase 3 (6-7 months):

Investigation of available metrics to measure complexity keeping in mind usability (focusing on its two sub characteristics: communicativeness and understandability) of software product line feature model.

Propose improvement/ new metrics to measure complexity.

Phase 4 (4-10 months):

Theoretical validation of improvement/ new metrics and through controlled experimentation.

Phase 5 (4-5 months):

Analyze the result obtained and thesis writing.

6.2. Gantt chart:

Project Plan	Months											
	2	4	6	8	10	12	14	16	18	20	22	24
Phase 1												
Phase 2												
Phase 3												
Phase 4												
Phase 5												

Figure 3 Gantt chart

6.3. Methodology:

- Research Begins.
- Study of Software Product Line Feature Model, UML Model, and Domain Engineering.
- Study of simple and feature oriented structural metrics.
- To analyze whether metrics are available to measure structural complexity?
- If yes, then find limitations in existing metrics and propose improvement for increased efficiency.
- If no, then propose new metrics to measure structural complexity, keeping in mind usability of SPL feature model.
- Theoretical Validation of the proposed metrics and through controlled experimentation.
- Draw conclusion and result generation.

Research begins
Study of Software Product Line Feature Model, UML Model, and Domain Engineering
Study of simple and feature oriented structural metrics
Are metrics available to measure structural complexity?
No
Yes
Find limitations in existing metrics
Propose new metrics to measure structural complexity of SPL feature model
Propose improvement for increased efficiency
Validation of the proposed metrics
Drawing conclusions
Result Generation

Figure 4 Flowchart depicting methodology adopted

7. Place of work and facilities available

To follow the line of investigation the research will be carried out in the laboratories existing in the university. The hardware and software available here will be used as a part of the research work.

8. References, bibliography and webliography

[1] S.R. Schach and A. Tomer, "Development/Maintenance/Reuse: Software Evolution in Product Lines", the Springer International Series in Engineering and Computer Science, Volume 576, 2000, Pages 437-450.

[2] I.Sommerville, Software Engineering, Pearson Education, India, 2008.

[3] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley, Boston, 2001.

[4] G. K. Hanssen, "Opening Up Software Product Line Engineering", Proceedings of the 2010 ICSE Workshop on Product Line Approaches in Software Engineering 2010, ACM, New York, USA, Pages 1-7.

[5] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report Cmu/Sei-90-Tr-021 , 1990.

[6] S. Apel and C. Kastner, "An Overview of Feature-Oriented Software Development," J. Object Technology (JOT), Volume 8, No. 5, 2009, Pages 49-84.

[7] A. Sharma, S.S. Sarangdevot, "Investigating the Application of Feature-Oriented Programming in the Development of Banking Software Using Eclipse-Featureide Environment", International Journal of Computer Science & Technology, March 2011,Volume 2, Issue 1 ,Pages 53-57.

[8] D. B. Cuevas, "On The Automated Analysis of Software Product Lines using Feature Models-A Framework for Developing Automated Tool Support", The Department of Computer Languages and Systems, Spain, June 2007

[9] E. Bagheri, D. Gasevic, "Assessing the Maintainability of Software Product Line Feature Models Using Structural Metrics", Springer, September 2011, Volume 19, Issue 3, Pages 579-612.

[10] N.Fenton, S.L.Pfleeger, "Software Metrics: A Rigorous and Practical Approach", 2nd Ed., International Thomson Computer Press, 1996.

[11] D. Beuche, M. Dalgarno, "Software Product Line Engineering with Feature Models". Available at- http://www.pure-systems.com/fileadmin/downloads/pure-variants/tutorials/SPLWithFeatureModelling.pdf.

[12] S. Montagud, S. Abraha˜O. E. Insfran, "A Systematic Review of Quality Attributes and Measures for Software Product Lines", Software Quality Journal , September 2012, Volume 20, Issue 3-4, Pages 425-486.

[13] M. Svahnberg, J.Bosch, "Evolution in Software Product Lines", Journal of Software Maintenance: Research and Practice, November/December 1999, Volume 11, Issue 6, Pages 391–422.

[14] L. Chen, M. A. Babar, N. Ali, "Variability Management in Software Product Lines: A Systematic Review", Proceedings of the 13th International Software Product Line Conference, Pittsburgh, Pa, 2009, Pages 81-90.

[15] von der Maßen, T., H. Lichter, "Deficiencies in Feature Models", Workshop on Software Variability Management for Product Derivation-towards Tool Support. August 2004.

[16] S.M.Jamali, "Object Oriented Metrics", Survey Paper, 2006, Sharif University of Technology, Tehran Iran.

[17] J. K. Kearney, R.L. Sedlmeyer, W. B. Thompson, M. A. Gray, and M.A. Adler, "Software Complexity Measurement", Communications of the ACM, November 1986, Volume 29, Issue 11, Pages 1044 – 1050.

[18] E. A. Oliveira Junior, J.C. Maldonado, I.M. S. Gimenes**, "**Empirical Validation of Complexity and Extensibility Metrics for Software Product Line Architectures", Proceedings of the 2010 Fourth Brazilian Symposium on Software Components, Architectures and Reuse, 2010, Pages 31-40.

[19] Asim Rahman, "Metrics for the Structural Assessment of Product Line Architecture", Master Thesis, School of Engineering Blekinge Institute of Technology, Sweden, Thesis no: MSE-2004:24, August 2004.

[20] T. Zhang, L. Deng, J. Wu, Q. Zhou, and C. Ma, "Some Metrics for Accessing Quality of Product Line Architecture" in CSSE '08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering. Washington, Dc, USA, IEEE Computer Society, 2008, Pages 500–503.